Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aljaž Markežič

Napovedovanje odpovedi stroja za proizvodnjo tesnil z metodami strojnega učenja

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Zoran Bosnić

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kandidat naj v diplomski nalogi obravnava problem napovedovanja odpovedi industrijskega stroja. Modeliranja odpovedi naj se loti kot problema nadzorovanega strojnega učenja, preizkusi pa naj različne algoritme za napovedno modeliranje. Uspešnost algoritmov naj primerja tudi z nevronskimi mrežami z elementi LSTM (Long Short-Term Memory). Implementirane algoritme naj primerja in ovrednoti.

Zahvaljujem se mentorju izr. prof. dr. Zoranu Bosniću za nasvete in pomoč, ki mi jih nudil med izdelavo diplomske naloge, ter za njegovo neizmerno potrpežljivost in predanost.

Posebna zahvala gre še Marku Zadravcu, ki mi je priskrbel podatke in nudil dodatno pomoč pri izdelavi diplomske naloge.

Nenazadnje se zahvaljujem vsem, ki so mi nudili povratne informacije glede diplomske naloge in mi tako pomagali pri izboljšavi končnega izdelka.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
NN	neural network	nevronska mreža
RNN	recurent neural network	rekurenčna nevronska mreža
SVM	support vector machine	metoda podpornih vektorjev
LR	logistic regression	logistična regresija
RF	random forest	naključni gozd
LSTM	long short-term memory	nevronske mreže z dolgim kratkoročnim spominom
ARIMA	autoregressive integrated moving average	integrirani avtoregresijski model z gibajočim povprečjem
GRU	gated recurent unit	nevronske mreže z uteženo rekurenčno enoto
AUC	area under the ROC curve	ploščina pod krivuljo ROC
VAR	vector autoregression	vektorska avtoregresija
MLP	multilayer perceptron	večnivojski perceptron
KNN	k nearest neighbors	k najbližjih sosedov
GLM	generalized linear model	posplošeni linearni model
ARMA	autoregressive moving average	avtoregresijski modeli z gibajočim povprečjem
ROC	receiver operating characteristic	karakterisika delovanja sprejemnika

Povzetek

Naslov: Napovedovanje odpovedi stroja za proizvodnjo tesnil z metodami strojnega učenja

Cilj diplomske naloge je izdelava sistema za napovedovanje odpovedi industrijskih strojev v sodelujočem podjetju. Reševanja problema smo se lotili z uporabo pristopov in metod strojnega učenja. Najprej smo se lotili preoblikovanja in filtriranja podatkov, da jih lahko uporabimo kot učne podatke za klasifikacijske modele, nato smo izvedli učenje klasifikatorjev in ovrednotili uspešnost modelov. V prvem pristopu smo uporabili samo nabor zadnjih časovnih podatkov in modele brez sposobnosti pomnjenja; v drugem pristopu smo uporabili zgodovinsko obogateno množico podatkov; v tretjem pristopu pa smo uporabili nezgodovinske podatke z modeli, ki imajo sposobnost pomnjenja (LSTM in GRU). Na podlagi rezultatov smo ugotovili, da je tretji pristop najbolj uspešen.

Ključne besede: strojno učenje, klasifikacija, časovne vrste, napovedovanje, napoved odpovedi.

Abstract

Title: Predicting failure of rubber seal production machine using machine learning methods

The goal of the thesis is implementation of a predictive system for detecting failures in industrial machines. We tackle the problem by using different machine learning approaches and methods. Initially, we transformed the received data into a representation for supervised learning. In the next step we trained the classifiers and evaluated their performance. We applied three different approaches, as follows. In the first approach we trained memoryless models without using historical data; in the second approach we extended the data with additional historical attributes; in the third approach we trained memory-retaining models (LSTM and GRU) with a non-historic dataset. On the basis of our experimental results we discovered that the third approach gives as the best results.

Keywords: machine learning, classification, time sequence, forecasting, failure prediction.

Poglavje 1

Uvod

V gospodarstvu je proces vzdrževanja in popravila industrijskih strojev zelo drag in zamuden proces. Okvare industrijskih strojev povzročajo zaustavitve proizvodnih procesov in posledično dodatne stroške. V ta namen so se v proizvodnjo vpeljali različni postopki za diagnozo strojne opreme, ki skušajo na podlagi razumevanja preteklih odpovedi stroja napovedati delovanje stroja v prihodnosti. Primera takih postopkov sta zaznavanje nenavadnih vibracij na elektromotorjih [13] ali uporaba Weibullove porazdelitve za napovedovanje odpovedi hladilne črpalke v hidroelektrarni [16].

Zaradi omejitev zmogljivosti hranjenja podatkov, računske moči in pristopov za obdelavo podatkov je bila v preteklosti uporaba zahtevnejših pristopov za diagnozo oziroma napovedovanje okvar strojev zelo redka. Z odpravo teh omejitev se je zelo povečala uporaba metod strojnega učenja (angl. machine learning) in umetne inteligence (angl. artificial inteligence), ki uporabljajo procesno in spominsko zelo zahtevne postopke.

Danes je v industrijskem okolju zelo običajno shranjevanje velikih količin podatkov, ki jih med delovanjem stroja generirajo senzorska omrežja. Shranjeni podatki se večinoma uporabljajo kot dnevnik (angl. log) delovanja stroja za vzdrževalne namene. Zaradi dokazanih izboljšav, ki so jih prinesle metode **strojnega učenja** in **umetne inteligence** na številnih področjih, so nekatera podjetja začela vlagati v raziskovanje uporabnosti teh metod v

industrijskem okolju.

Namen diplomske naloge je raziskati uporabnost pristopov **strojnega učenja** pri reševanju problema napovedi okvar strojev. V naslednjem razdelku bomo podrobneje opisali problem in cilj diplomske naloge.

1.1 Opis problema

Podjetje, s katerim sodelujemo za potrebe raziskav, se ukvarja s proizvodnjo gumijastih obročkov in tesnil. Proizvodni cikel je sestavljen iz človeškega in strojnega dela. Proizvodni cikel poteka tako, da delavec sproži stikalo na stroju, kalup se zapre, vanj se vbrizga tekoča guma, ki se nato zapeče in izsuši. Ko se ta del procesa zaključi, se kalup odpre, delavec vzame gumijasti obroček iz kalupa, tega očisti in praznega vrne nazaj v stroj. Opisani postopek se nato ponavlja ciklično.

V idealnih pogojih bi proces proizvajanja gumijastih obročkov potekal neprekinjeno. Ker pa pogoji niso takšni, se na strojih občasno pojavljajo okvare, ki prekinejo proizvodni proces. V primeru prekinitve mora podjetje poiskati zunanjo pomoč strokovnjaka, ki stroj na novo konfigurira in proizvodni proces znova zažene. Podatki o delovanju/nedelovanju stroja in mnogi drugi parametri procesa se med proizvodnim procesom shranjujejo v podatkovno bazo.

Cilj diplomske naloge je raziskati uporabnost pristopov strojnega učenja in izdelati sistem za napovedovanje okvar strojev. Zaradi obširnosti področja strojnega učenja se bomo v diplomski nalogi omejili na metode in pristope, ki so opisani v naslednjem poglavju.

Poglavje 2

Sorodna dela in metodologija

2.1 Sorodna dela

V literaturi je prisotnih nekaj člankov, ki rešujejo problem napovedovanja okvare strojev z različnimi metodami strojnega učenja.

Kolokas in sod. [10] opisujejo podoben problem. Avtorji so poskusili uporabiti različne metode strojnega učenja pri napovedi delovanja stroja za litje aluminija za 5, 10 in 45 minut vnaprej. Stroj je opremljen z različnimi senzorji, od katerih dobimo 29 atributov, ki opisujejo trenutno stanje stroja. Za napovedovanje so uporabili sledeče modele: **naključni gozd** (angl. random forest), **odločitvena drevesa** (angl. decision trees), **MLP**, **naivni Bayes**, **linearno regresijo**, **SVM** in **RNN**. Poleg vseh uporabljenih metod učenja so implementirali še klasifikator, ki je ne glede na vrednosti atributov napovedoval nedelovanje stroja. Če si v raziskovalnem delu ogledamo rezultate, kjer so izbrani najboljši modeli, lahko pri napovedi za 45 minut vnaprej opazimo, da je točnost napovedi zelo visoka (približno 95 %), občutljivost pa relativno nizka (približno 60 % za določene modele). Občutljivost (angl. sensitivity ali recall) nam pove razmerje med pravilno in nepravilno napovedanimi pozitivnimi razredi (v našem primeru je nedelovanje pozitivni razred). V praksi to pomeni, da je naša napoved za nedelovanje stroja nezanesljiva. Napovedi za 5 in 10 minut vnaprej so boljše, vendar avtorji opozarjajo, da je to časovno

obdobje prekratko za preprečitev hujših okvar na stroju. V zaključku avtorji omenijo, da bi bilo vredno poskusiti problem rešiti s pristopom RNN.

Larky in sod. [13] obravnavajo napovedovanje odpovedi elektromotorjev v industrijskih strojih na podlagi meritev vibracij elektromotorja. Za razliko od prvega članka je ta veliko bolj usmerjen, saj ne primerja različnih modelov, ampak se osredotoči samo na uporabo metode nevronskih mrež. Vhodni podatki predstavljajo vrednosti senzorja, ki spremlja intenzivnost vibracij motorja s časovnim intervalom 5 sekund. Avtorji se problema lotevajo tako, da vhodne podatke najprej pošljejo skozi algoritem Kaplan-Meier in gostotno funkcijo za napoved degradacije. Izhodne podatke prejšnjih algoritmov nato uporabijo kot vhodne podatke nevronske mreže. Rezultati metode nevronskih mrež predstavljajo verjetnosti, da se elektromotor okvari v naslednjih 5 serijah po 5 sekund (torej 5, 10, 15, 20, 25 sekund vnaprej). Če verjetnost katerekoli serije pade pod prag 50 %, jemljemo to kot okvaro stroja. Avtorji poročajo, da model deluje s točnostjo nad 96 %. V zaključku še omenijo, da je uporaba nevronskih mrež za napoved okvar strojev zelo uporabna, saj zmanjša potrebe po intervenciji vzdrževalcev.

Zhang in sod. [16] obravnavajo napovedovanje stanja hladilne črpalke v elektrarni z uporabo modela **LSTM** (angl. Long short-term mermory) in njegovo primerjavo z modelom **ARIMA**. V uvodu avtorji sklepajo, da bo LSTM deloval boljše zaradi lastnosti kratkoročnega spomina in selektivnega pomnjenja, ki mu omogoča, da si zapomni samo "pomembne" informacije. Za razliko od **LSTM** je ARIMA veliko preprostejši, saj nima lastnosti selektivnega spomina. Črpalka vsebuje 33 senzorjev, katerih vrednosti se shranjujejo v podatkovno bazo. Avtorji članka so poskušali z uporabo modela LSTM napovedati vrednosti posameznega senzorja za določeno časovno enoto vnaprej. 67 % količine vseh podatkov je bilo uporabljenih za treniranje modela, 33 % pa za testiranje. Rezultati primerjave med dejanskimi in napovedanimi vrednostmi so bili zadovoljivi, saj je model LSTM vrednost vsakega senzorja napovedal bolje kot model ARIMA (povprečen ARIMA-RMSE 0.080, LSTM-RMSE 0.020).

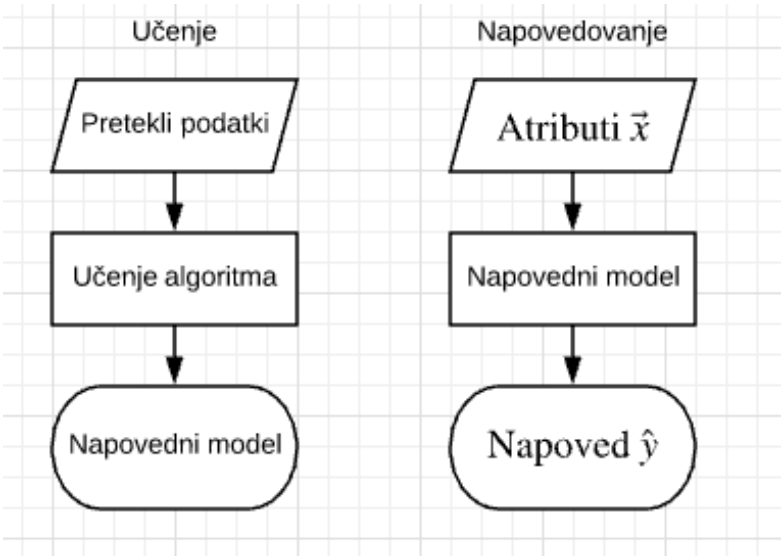
Baptista in sod. [1] opisujejo problem napovedovanja okvare ključnega ventila, ki je del ventilacijskega sistema letalskega plovila. Ventil je izpostavljen visokim vibracijskim, sevalnim in toplotnim obremenitvam, kar sproži proces pospešene degradacije ventila. Po besedah avtorjev je ventil pogost vzrok za prerazporejanje letalskih flot. Z napovedovanjem povprečnega časa med okvarami (angl. MTBF - mean time between failure) je možno zmanjšati stroške, ki nastanejo pri prerazporejanju letov in številnih nepotrebnih popravilih ventila. Problema se lotevajo tako, da vhodne podatke za določeno časovno obdobje pošljejo skozi model ARMA in izračunajo statistične kazalnike, kot so standardna deviacija, minimum, maksimum, kurtosis ... Dobljene podatke so nato združili v vektorje, ki so jih uporabili kot vhodne podatke za različne modele strojnega učenja (NN, KNN, RF, GLM, SVM). Rezultate so primerjali s tradicionalnejšim pristopom napovedovanja odpovedi strojev, ki uporablja Weibullov model. Ugotovili so, da najboljše rezultate ponuja model SVM. Prav tako so dokazali, da metode strojnega učenja boljše napovedujejo življenjsko dobo opreme kot tradicionalnejši pristopi.

2.2 Metode strojnega učenja

2.2.1 Klasifikacija

Klasifikacija je proces določanja razredov (lastnosti) primerom vzorca populacije na podlagi učne množice, primeri katere že imajo določene razrede. En najbolj znanih primerov klasifikacije je filtriranje elektronskih sporočil, kjer prihajajočim sporočilom določimo razred zaželen (angl. spam) ali nezaželen (angl. no-spam) glede na oznako preteklih elektronskih sporočil. Ker klasifikacija potrebuje primere z že določenimi klasifikacijskimi razredi, spada v področje **nadzorovanega učenja** (angl. supervised learning). Rezultat nadzorovanega učenja je funkcija, ki neklasificiranim primerom določi razred. Za lažje razumevanje postopka **učenja** in **napovedovanja** imamo na sliki 2.1 prikazan shematski potek obeh postopkov.

Rezultat klasfikacijskega modela je funkcija $\hat{y} = f(\vec{x})$, kjer $\hat{y}$ predstavlja



Slika 2.1: Vizualizacija procesa nadzorovanega učenja in napovedovanja

napovedan razred, $\vec{x}$ pa vhodne atribute. Cilj učenja modela je pridobiti tako funkcijo, kjer je klasifikacijska napaka čim manjša.

V naslednjih razdelkih so predstavljeni modeli, ki smo si jih izbrali za reševanje problema napovedovanja okvar strojev.

2.2.2 Logistična regresija

Logistična regresija se pogosto uporablja pri klasifikacijskih problemih. Natančneje se uporablja pri reševanju problemov **binarne klasifikacije**, kjer imamo samo dva klasifikacijska razreda, npr. ["*ne_deluje*","*deluje*"]. Logistična regresija nam poda verjetnost, da določen vzorec pripada enemu od dveh klasifikacijskih razredov. Čeprav je naš problem nelinearen, je metoda logistične regresije še vedno pogosto uporabljena zaradi svoje hitrosti pri učenju in preprostosti delovanja. Logistična regresija je dobila svoje ime po sigmoidni (logistični) funkciji, ki se uporablja kot prag za določanje pripadnosti vzorca.

2.2.3 Naključni gozdovi

Naključni gozdovi (angl. random forest) oziroma naključni odločitveni gozdovi so eden izmed algoritmov nadzorovanega učenja in spadajo pod pristop učenja ansamblov (angl. ensamble learning) [6]. Ta pristop temelji na predpostavki, da je kombinirano znanje več **preprostih modelov** boljše kot znanje enojnega **kompleksnejšega modela**. Preprostejše modele dobimo bodisi tako, da jih izpostavimo podmnožici učnih podatkov, ali z uporabo različnih parametrov učnega algoritma. Naključni gozd je sestavljen iz množice odločitvenih dreves (angl. decision trees), ki so bila izpostavljena podmnožici učnih podatkov. Napovedovanje se odvija prek preprostega glasovalnega sistema, kjer vsako drevo glasuje za razred, ki se mu zdi na podlagi pridobljenega znanja najverjetnejši. Izhodni razred naključnega gozda je enak večinskemu razredu glasov odločitvenih dreves. Pristop je zanimiv, ker imamo pri njem namesto običajne metode, kjer imamo en zelo močan (vseved) klasifikator, več šibkih (specializiranih) klasifikatorjev, ki se lahko boljše prilagajajo določenim specifikam podatkov. Posamezna odločitvena drevesa učimo tako, da se za vsako notranje vozlišče dreves upošteva le naključna podmnožica atributov učne množice in se zgradi odločitveno drevo po standardnem postopku (npr. glede na informacijski prispevek). Naključni gozdovi poleg zgoraj opisanega pristopa izbire atributov uporabljajo še metodo *bagging*, ki model stabilizira, izboljša napovedno točnost in zmanjša varianco.

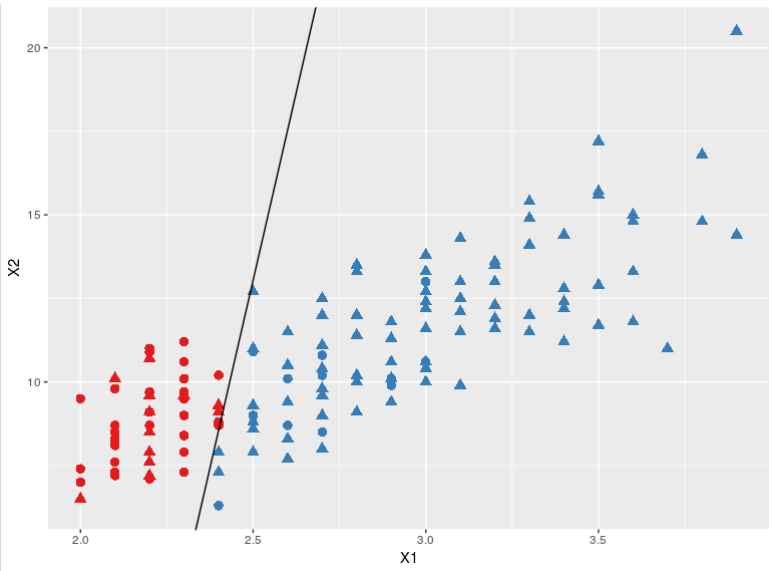
2.2.4 Metoda podpornih vektorjev

Metoda podpornih vektorjev oziroma SVM je široko uporabljen pristop za reševanje klasifikacijskih problemov [3]. Osnovna ideja algoritma je relativno preprosta. Če vzamemo za primer binarno klasifikacijo, je naš cilj, da najdemo hiperravnino (v primeru binarne klasifikacije je to premica), ki najboljše ločuje klasifikacijska razreda med seboj. To dosežemo tako, da minimiziramo razdaljo med primeri istega razreda. Če taka hiperravnina

obstaja, pravimo, da je problem **linearno ločljiv**. Primere, ki so najbližje hiperravnini, imenujemo **podporni vektorji**.

Cilj algoritma SVM je maksimiziranje razdalje med podpornimi vektorji in hiperravnino. Če imamo večdimenzionalen problemski prostor, uporabimo tako imenovane **jedrne funkcije** (angl. kernels), ki nam časovno učinkovito preslikajo prvotni problemski prostor v drugačnega. Glavni razlog za uporabo jedrnih funkcij (polinomska funkcija, radialna funkcija, sigmoidna funkcija) je preslikava problema v problemski prostor, kjer je problem lažje rešljiv.

Slika 2.2 prikazuje delovanja algoritma SVM na problemu binarne klasifikacije. Modra barva prikazuje vse primere, ki so bili klasificirani kot delovanje stroja, rdeča pa primere, ki so bili klasificirani kot nedelovanje stroja. Premica, ki ločuje modre in rdeče primere, predstavlja **hiperravnino**, ki najboljše ločuje klasifikacijska razreda.



Slika 2.2: Gragični prikaz delovanja metode podpornih vektorjev

2.2.5 Long Short-term Memory

Long Short-term Memory je eden izmed gradnikov, ki se uporabljajo za gradnjo rekurentnih nevronskih mrež (RNN) [7]. Preden definiramo LSTM, se moramo spoznati s pojmom rekurenčnih nevronskih mrež.

Rekurenčne nevronske mreže so posebna oblika nevronskih mrež, kjer povezave med posameznimi nevroni predstavljajo **usmerjen graf**. Za razliko od običajnejših konvolucijskih nevronskih mrež, kjer podatki potekajo od vhodnih vozlišč, prek več slojev skritih plasti do izhodnih vozlišč, so lahko povezave pri RNN usmerjene tudi v "nasprotno" smer. Ta značilnost jim poda lastnost pomnjenja in s tem boljšo sposobnost učenja **zaporedij** in **časovno odvisnih podatkov**.

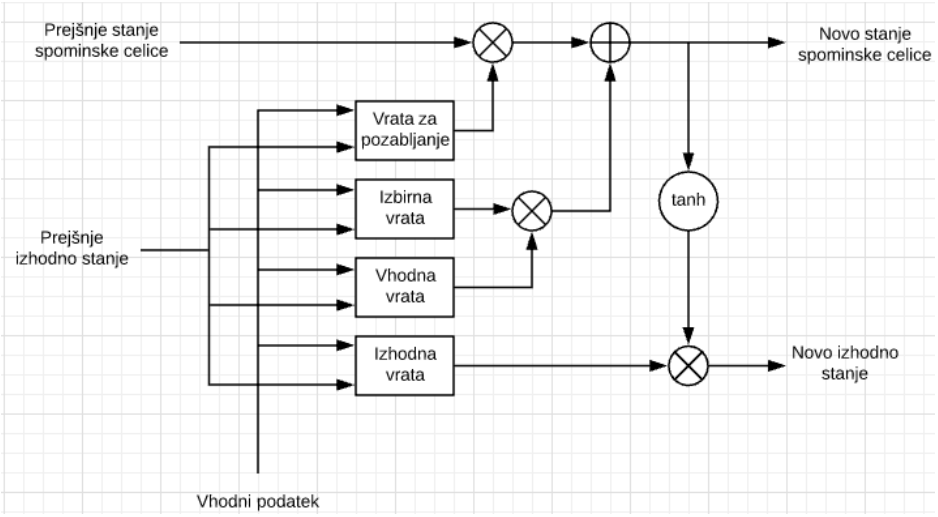
LSTM je specifičen primer gradnika rekurenčnih nevronskih mrež. Običajen gradnik LSTM je sestavljen iz naslednjih delov:

- **spominska celica:** začasno shranjuje vhodne podatke,
- **vrata za pozabljanje:** nadzorujejo podatke v spominski celici (kaj si zapomniti in kaj pozabiti),
- **izbirna vrata:** nadzorujejo pretok podatkov v spominsko celico,
- **izhodna vrata:** nadzorujejo pretok podatkov iz gradnika LSTM,
- **vhodna vrata:** nadzorujejo pretok podatkov v gradnika LSTM.

Zgoraj opisana struktura gradnika LSTM je prikazana na sliki 2.3.

2.2.6 Gated recurrent unit

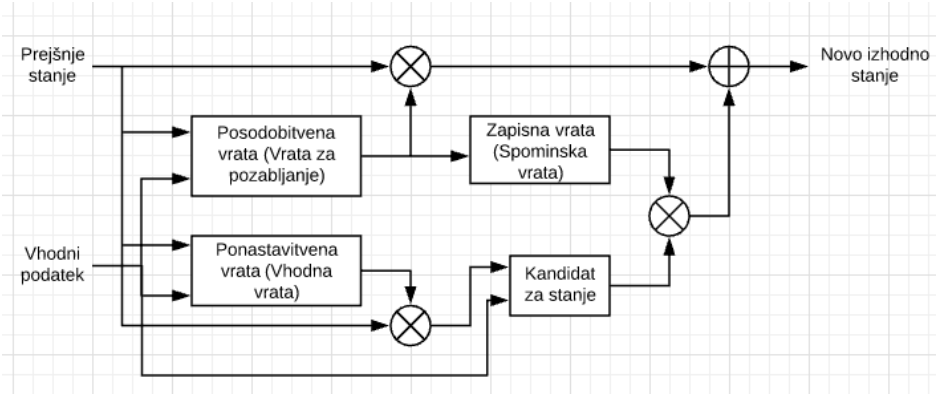
Gated recurrent unit je poenostavljena različica gradnika **LSTM**, ki jo je leta 2014 predlagal Kyunghyun Cho [4], z namenom razrešitve problema **izginjajočega gradienta** (angl. vanishing gradient problem), ki nastane pri gradientnih pristopih učenja (angl. gradient-based learning methods).



Slika 2.3: Shema zgradbe gradnika LSTM

Problem izginjajočega gradienta nastane zaradi vzvratnega razširjanja napake (angl. backpropagation), ki se uporablja med postopkom učenja nevronske mreže. Postopek vzvratnega razširjanja napake uporablja verižno pravilo odvajanja, ki množi odvode. Ker se vrednosti odvodov pogosto nahajajo med 0 in 1, se z daljšimi verigami množenja vrednost produkta manjša v nedogled (dokler ne doseže vrednosti 0). To povzroči, da se začetni sloji nevronske mreže učijo počasneje od poznejših. Začetni sloji nevronske mreže so zadolženi za prepoznavo preprostih vzorcev v podatkih, ki so temeljni gradniki za poznejše sloje, in v končni fazi za dobro uspešnost modela.

GRU se od **LSTM** razlikuje v odsotnosti izhodnih vrat, ki so pri LSTM-ju zadolžene za nadzor pretoka podatkov iz spominske celice v preostale dele nevronske mreže, kar je prikazano na sliki 2.4. Ker GRU tega mehanizma nima, je celotna vsebina spominske celice izpostavljena kot izhodni podatek v naslednji gradnik. Uporablja se jih na istih področjih kot LSTM, torej za modeliranje zvoka, prepoznavanje govora in pri podobnih problemih, ki zahtevajo prepoznavo zaporedij. Poleg preprostejše implementacije ima GRU lastnost boljše uspešnosti pri učenju na manjših učnih množicah.



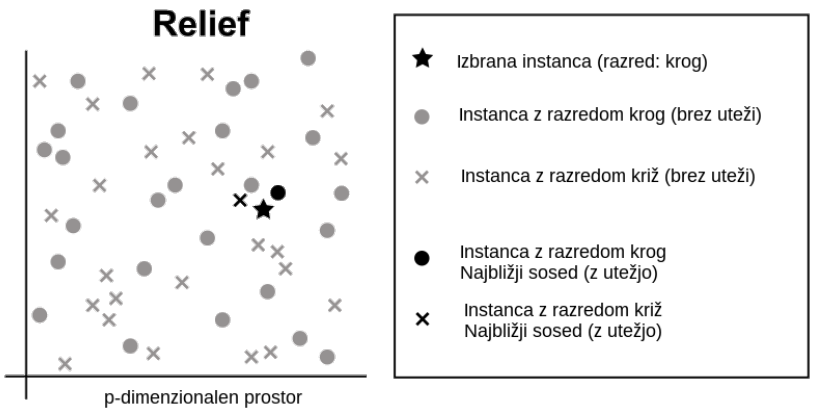
Slika 2.4: Shema zgradbe gradnika GRU

2.3 Izbira atributov z algoritmom Relief

Relief je algoritem, ki sta ga leta 1992 razvila Kenji Kira in Larry A[9]. Zasnovan je bil z namenom uporabe pri problemih **binarne klasifikacije** z diskretnimi ali zveznimi atributi. Glavna značilnost algoritma je ocenjevanje atributov na podlagi interakcij z drugimi atributi. Relief ocenjuje atribute tako, da spremlja razlike med vrednostmi atributov v parih primerov učne množice. Pari so sestavljeni iz izbranega primera v učni množici in njenega najbližjega soseda. Če sta pripadnika para iz istega razreda (zadetek), se ocena opazovanega atributa zmanjša tako, da se ji odšteje razlika vrednosti para. Če sta pripadnika para iz različnega razreda (zgrešitev), se ocena atributa poveča tako, da se razlika vrednosti para prišteje h končni oceni atributa. Rezultat algoritma je vektor ocen posameznih atributov. Na sliki 2.5 je prikazano delovanje algoritma Relief.

2.4 Mere uspešnosti pri klasifikaciji

Eden najpomembnjših korakov pri strojnem učenju je ocena uspešnosti naučenega modela. Običajno se to izvede tako, da se modelu poda množico podatkov, na kateri nismo učili modela. Napovedane klasifikacijske razrede

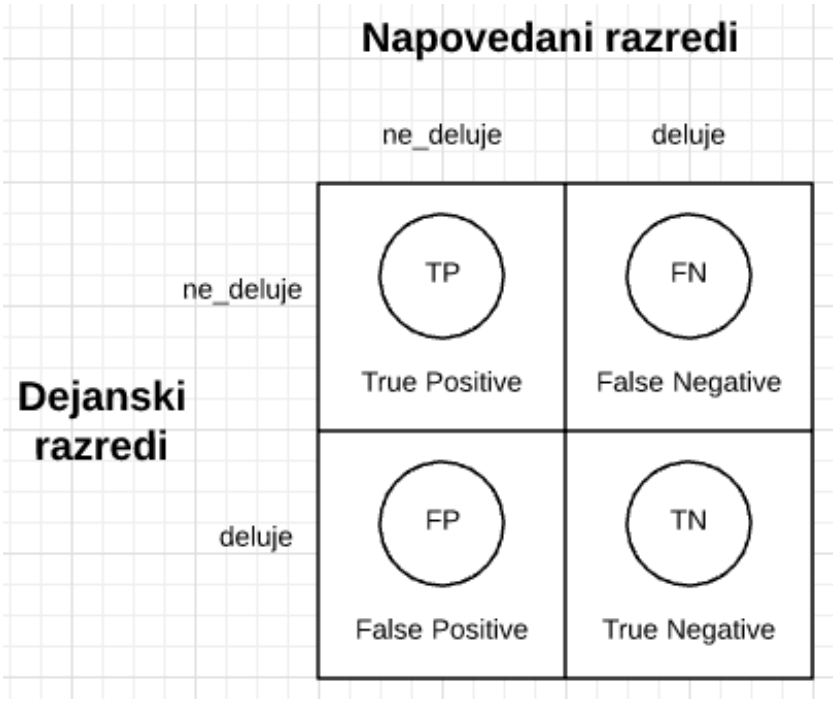


Slika 2.5: Vizualizacija delovanja algoritma Relief. V koordinatnem sistemu imamo prikazano vizualizacijo učne množice, kjer so z krogi predstavljeni pripadniki enega razreda, s križci pa pripadniki drugega razreda. Poudarjena primera posameznega razreda predstavljata najbližja soseda opazovanega primera, ki je označen z zvezdo.

se nato primerja z dejanskimi razredi primerov testne množice. Običajen pristop je, da zgradimo **klasifikacijsko matriko** (angl. confussion matrix) in s kombinacijo različnih vrednosti matrike izračunamo dodatne metrike, ki predstavljajo uspešnost modela.

Na področju strojnega učenja je **klasifikacijska matrika** način vizualizacije klasifikacijske uspešnosti modela. Največkrat se uporablja pri problemih binarne klasifikacije, kjer imamo samo dva razreda. Občasno se uporablja tudi za n-dimenzionalne probleme, kjer dobi matrika obliko $n * n$. Pri binarnih problemih ima značilno obliko 2×2 matrike, ki nam prikazuje, kolikokrat se je napovedni model zmotil/zadel pri napovedi klasifikacijskega razreda. Videz binarne klasifikacijske matrike je prikazan na sliki 2.6. V našem problemu napovedovanja okvare stroja definirajmo z *"ne deluje"* kot pozitivni razred in z *"deluje"* kot negativni razred. Klasifikacijska matrika na sliki 2.6 prikazuje naslednje primere:

- **true positive (TP)**: število pravilno klasificiranih primerov pozitiv-



Slika 2.6: Struktura klasifikacijske matrike pri binarni klasifikaciji

nega razreda,

- **true negative (TN)**: število pravilno klasificiranih primerov negativnega razreda,

- **false positive (FP)**: število nepravilno klasificiranih primerov pozitivnega razreda,

- **false negative (FN)**: število nepravilno klasificiranih primerov negativnega razreda.

S kombinacijo zgoraj opisanih vrednosti lahko dosežemo več različnih metrik, ki ocenjujejo uspešnost modela. Te opisujemo v naslednjih razdelkih.

2.4.1 Klasifikacijska točnost

Klasifikacijska točnost je definirana kot razmerje med celotno testno množico in pravilno napovedanimi primeri [12]:

$$\text{klasifikacijska točnost} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Iz klasifikacijske točnosti lahko izpeljemo še nasprotno vrednost, to je **mero napake**, ki se jo izračuna po naslednji formuli:

$$\text{mera napake} = 1 - \frac{TP + TN}{TP + TN + FP + FN} = 1 - \text{točnost} \tag{2.2}$$

2.4.2 Občutljivost

Občutljivost (angl. sensitivity) nam pove razmerje pravilno klasificiranih primerov pozitivnega razreda [12]. Izračuna se kot razmerje med številom pravilno klasificiranih pozitivnih primerov in številom vseh primerov, ki jim je bil dodeljen pozitivni razred (vključuje tudi nepravilne klasificirane primere).

$$\text{občutljivost} = \frac{TP}{TP + FN} \tag{2.3}$$

Pri reševanju našega problema nam je najpomembnejša občutljivost, saj to kaže na uspešnost modela pri zaznavanju okvar strojev.

2.4.3 Specifičnost

Specifičnost (angl. specificity) nam pove odstotek pravilno klasificiranih primerov negativnega razreda [12]. Izračuna se kot razmerje med številom pravilno klasificiranih negativnih primerov in številom vseh primerov, ki jim je bil dodeljen negativni razred (vključuje tudi nepravilno klasificirane primere).

$$\text{specifičnost} = \frac{TN}{TN + FP} \tag{2.4}$$

2.4.4 F-ocena

F-ocena je mera uspešnosti, ki kombinira dve drugi meri, preciznost (natančnost) in priklic (občutljivost) [12]. Meri kombiniramo tako, da uporabimo harmonično povprečje, pri katerem se računa z razmerji. V našem primeru je to ugodno, saj sta tako točnost kot priklic podana kot razmerji (zaloga vrednosti med [0,1]). Priklic oziroma **občutljivost** sem opisal že v prejšnjem razdelku. Preciznost (natančnost) pa je podana kot:

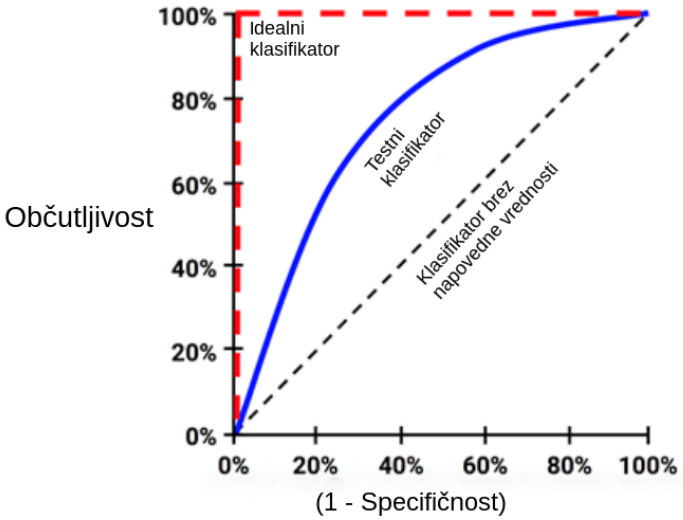
$$\text{preciznost} = \frac{TP}{TP + FP} \tag{2.5}$$

$$\text{F-ocena} = \frac{2 * \text{preciznost} * \text{občutljivost}}{\text{občutljivost} + \text{preciznost}} = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.6}$$

2.4.5 AUC - ploščina pod krivuljo ROC

Krivulja ROC (angl. receiver operating characteristic) je pogosto uporabljena kot pristop za analizo kompromisa med prepoznavo pravilnih pozitivnih razredov in izogibanjem napačne klasifikacije v pozitivne razrede [12]. Krivulja ROC je bila razvita med drugo svetovno vojno z namenom ocene uspešnosti sprejemnika pri ločevanju pravilnih signalov od lažnih. Metoda je danes uporabna za vizualizacijo uspešnosti klasifikacijskih modelov. Točke, ki sestavljajo ROC, prikazujejo mero pravilnih pozitivnih razredov (angl. true positive) pri spreminjajoči se meri nepravilnih pozitivnih razredov (angl. false positive). Na osi x imamo delež lažnih pozitivnih primerov, na osi y pa delež pravilnih pozitivnih primerov. Ker so te mere sorodne meram občutljivosti in specifičnosti, predstavlja os x (1-specifičnost), os y pa občutljivost.

Če želimo s to krivuljo dobiti mero uspešnosti modela, lahko merimo **AUC** (angl. area under the curve), kar je ploščina območja pod krivuljo ROC [5]. Zaloga AUC je enaka [0-1]. Višja kot je vrednost, bolj se krivulja ROC prilega rdeči krivulji ROC, ki je prikazana na sliki 2.7.



Slika 2.7: Grafični prikaz krivulje ROC [12]

Poglavje 3

Priprava podatkov

3.1 Opis podatkov

V razdelku 1.1 smo omenili, da so stroji opremljeni s senzorji, katerih vrednosti shranjujemo v podatkovno bazo. Stroji so opremljeni z dvema vrstama senzorjev: **digitalnimi** in **analognimi**. Digitalni senzorji proizvajajo diskretne podatke z zalogo vrednosti [0,1], medtem ko analogni senzorji proizvajajo zvezne podatke z zalogo vrednosti $\mathbb{R} \geq 0$. Posamezen stroj je opremljen s 16 digitalnimi senzorji in z 8 analognimi. V tabeli 3.1 smo za vsak senzor zapisali tip, identifikacijsko številko in kratek opis senzorja.

Posamezen senzor spremlja stanje določene lastnosti stroja, kot je na primer temperatura vbrizgane gume. Ob spremembi spremljane vrednosti se vrednost senzorja zapiše v podatkovno bazo. Ker imamo več različnih senzorjev, ki oddajajo različne tipe signalov ob različnih trenutkih, si moramo v bazo poleg vrednosti senzorja shraniti dodatne podatke.

- `ai/diIndex`: identifikacijska številka senzorja,
- `ai/diValueRaw`: surova vrednost senzorja,
- `time_stamp`: čas spremembe signala.

Na slikah 3.1 in 3.2 je prikazana struktura digitalnih/analognih podatkov senzorjev.

Tip senzorja	ID številka	Opis
Digitalni	0	Dvigovanje dodaj
Digitalni	1	Dvigovanje gor
Digitalni	2	Brizg vbrizgavanje
Digitalni	3	Brizg nazaj
Digitalni	4	Brizg zunaj naprave
Digitalni	5	Režim obratovanja (avtomatski)
Digitalni	6	Režim obratovanja (ročni)
Digitalni	7	Režim obratovanja (nastavitev)
Digitalni	8	Režim obratovanja (servis)
Digitalni	9	Visoki pritisk, zapri
Digitalni	10	Vakumski ventil
Digitalni	11	Plošče skupaj
Digitalni	12	Zapiralni cilinder, odpri
Digitalni	13	Visoki pritisk zunaj naprave
Digitalni	14	Ventil za polnjenje zunaj naprave
Digitalni	15	Dvoročnovarnostno krmiljenje
Analogni	0	Reguliranje tlaka
Analogni	1	Kontrola L2H
Analogni	2	Kontrola L3H
Analogni	3	Brizg
Analogni	4	Reguliranje količina
Analogni	5	Sistemski pritisk hidr. črpalka 1
Analogni	6	Zapiralni tlak
Analogni	7	Grelna plošča gore zunaj cone 1

Tabela 3.1: Seznam razpoložljivih senzorjev

	X1	id	diIndex	diMode	diValueRaw	time_stamp
1	1	915735	2	0	1	2018-03-13 23:08:47
2	2	915734	1	0	0	2018-03-13 23:08:47
3	3	915733	1	0	1	2018-03-13 23:08:44
4	4	915732	9	0	0	2018-03-13 23:08:44
5	5	915731	9	0	1	2018-03-13 23:08:39

Slika 3.1: Struktura podatkov digitalnih senzorjev v bazi

	X1	id	aiIndex	aiMode	aiValueRaw	time_stamp
1	1	105092934	7	0	15314	2018-03-13 23:11:19
2	2	105092933	6	0	41587	2018-03-13 23:11:19
3	3	105092932	5	0	1193	2018-03-13 23:11:19
4	4	105092931	4	0	3	2018-03-13 23:11:19
5	5	105092930	3	0	222	2018-03-13 23:11:19

Slika 3.2: Struktura podatkov analognih senzorjev v bazi

Tako oblikovanih podatkov ne moremo uporabiti kot vhodne podatke za klasifikacijske modele, saj ne predstavljajo celotnega stanja stroja v določenem trenutku. Poleg tega imamo zaradi beleženja sprememb vrednosti senzorja različno število podatkov za posamezen senzor. Klasifikacijski modeli zahtevajo, da so podatki oblikovani tako, da lahko iz posameznega učnega primera razberemo stanje stroja. Reševanja problema se bomo lotili s postopkom pretvorbe podatkov v vektorje, ki bodo predstavljali učne primere za nadzorovano učenje.

3.2 Oblikovanje učnih primerov

Pretvorbo senzorskih podatkov v učne primere smo izvedli z implementacijo funkcije **vectorize_data**. Funkcija deluje tako, da najprej generira zaporedje **enominutnih** časovnih intervalov. Iz na novo ustvarjenega zaporedja časovnih intervalov zgradi matriko, kjer predstavlja vsaka vrstica en časovni interval iz zaporedja. Nato se z zanko sprehodi skozi enakomerna časovna obdobja in v izvornih podatkih poišče vse vnose, ki se nahajajo v tem časovnem

obdobju. Vsak vektor z različnim identifikatorjem dodamo kot nov stolpec v novi matriki. Če vektor že obstaja, njegovo vrednost prištejemo k vsoti vseh njegovih vrednosti v tem časovnem obdobju. Ko pridemo do vnosa, ki se ne nahaja več v časovnem intervalu, povprečimo vsote vsakega. Če se določen atribut ni pojavil v tem časovnem obdobju, se mu dodeli vrednost atributa iz prejšnjega vektorja. Če prejšnjega vektorja ni (v primeru inicializacije matrike), se atributu dodeli vrednost 0. V algoritmu 1 je zapisana psevdokoda implementacije funkcije vectorize_data.

Algoritem 1: Psevdokoda funkcije vetorize_data

```
Generiranje enakomernih časovnih intervalov;
Inicializacija izhodne matrike;
Inicializacija izhodnega vektorja;
foreach časovni interval do
    Za vsak senzor poišči podatke v izvorni matriki;
    Vrednost senzorja prištej na pravo mesto v izhodnem vektorju;
    Povpreči vsote vrednosti senzorjev v vektorju s številom vrednosti
     senzorja za časovni interval;
    if V trenutnem vektorju ni vrednosti za določen senzor then
        if Ni prejšnje vrednosti za senzor then
            Nastavi vrednost senzorja na 0;
        else
            Nastavi vrednost senzorja na vrednost senzorja prejšnjega
             vektorja;
```

Rezultat funkcije je matrika, vidna na sliki 3.3, kjer vrstice predstavljajo stanje vseh senzorjev stroja za časovno obdobje ene minute. Tako oblikovane vhodne podatke lahko uporabimo kot vhodne podatke. V naslednjem razdelku se bomo lotili generiranja klasifikacijskega razreda za na novo ustvarjene učne primere.

moxa_1	moxa_2	moxa_3	moxa_4	moxa_5	moxa_6	moxa_7	moxa_8	moxa_9	moxa_10	moxa_11	moxa_12	moxa_13	moxa_14	moxa_15	moxa_21	moxa_22
0.5	0	0	1	0.5	0.5	0.5	0	0.5	0	1	0.5	0	0.5	0.5000000	1050.3394	1043.7697
0.5	0	0	0	0.5	0.5	0.5	0	0.5	0	1	0.5	0	0.5	0.5000000	1105.9167	1067.9345
0.5	0	0	0	0.5	0.5	0.5	0	0.5	0	0	0.5	0	0.5	0.6666667	949.5767	871.1350
0.5	1	0	0	0.5	0.5	0.5	0	0.5	0	1	0.5	0	0.5	0.0000000	535.8133	545.2530
0.5	1	0	0	0.5	0.5	0.5	0	0.5	0	1	0.5	0	0.5	0.0000000	629.0727	645.8667
0.5	1	0	0	0.5	0.5	0.5	0	0.5	0	1	0.5	0	0.5	0.0000000	942.2035	922.1327

moxa_23	moxa_24	moxa_25	moxa_26	moxa_27	time_start	time_end
864.0848	18376.212	6809.091	41378.18	15282.60	2018-03-13 23:05:38	2018-03-13 23:06:38
6018.4226	11154.583	3884.065	40815.64	15300.19	2018-03-13 23:06:38	2018-03-13 23:07:38
6836.4110	5899.319	3905.620	11421.38	15304.86	2018-03-13 23:07:38	2018-03-13 23:08:38
1603.6687	46779.380	31740.940	40341.44	15305.05	2018-03-13 23:08:38	2018-03-13 23:09:38
237.7333	33872.000	25463.558	42403.70	15312.87	2018-03-13 23:09:38	2018-03-13 23:10:38
226.8584	30275.177	22893.602	41801.85	15317.39	2018-03-13 23:10:38	2018-03-13 23:11:38

Slika 3.3: Nova struktura učne množice. Vsaka vrstica predstavlja vektor stanja stroja za obdobje ene minute

3.3 Določanje klasifikacijskega razreda

V prejšnjem razdleku smo prvotne ("surove") podatke preoblikovali v vektorsko obliko, ki je primerna za uporabo s klasifikacijskimi modeli. Naslednji korak priprave podatkov je določanje **klasifikacijskega razreda**, ki nam bo predstavljal delovanje stroja za n časovnih enot vnaprej.

Klasifikacijski razred bomo dobili z zamikom vrednosti atributa **moxa_5** za n časovnih enot vnaprej. Atribut moxa_5 predstavlja **delovanje** stroja za izbran časovni interval. Ker so vsi atributi časovno urejeni (od najstarejšega do najnovejšega), bomo zamik atributa moxa_5 dosegli s premikom vrednosti v stolpcu za n vrstic navzdol. S tem bomo dobili nov atribut **shifted**, ki predstavlja delovanje stroja za n časovnih enot vnaprej. Da je reševanje problema napovedi okvar bolj obvladljivo, smo si za n določili 3 vrednosti in s tem definirali tri različne napovedne probleme:

- n = 30 - napovedovanje delovanja stroja za 30 minut vnaprej,
- n = 60 - napovedovanje delovanja stroja za 1 uro vnaprej,
- n = 480 - napovedovanje delovanja stroja za 8 ur vnaprej.

Tako smo iz prvotne množice podatkov ustvarili 3 nove množice z dodatnim atributom shifted, ki predstavlja vrednost atributa moxa_5, zamaknjeno za na n časovnih enot vnaprej. Ker rešujemo klasifikacijski problem, moramo

zalogo vrednosti atributa shifted spremeniti iz zvezne množice v diskretno. Če je vrednost atributa shifted manjša od 1, pomeni, da stroj v opazovanem časovnem intervalu ene minute za kratek trenutek (krajši od minute) ni deloval. Ker je naš cilj napovedati okvaro oziroma nedelovanje stroja, je primer, ko stroj ni deloval za nek kratek čas ali ni deloval za celoten interval ene minute, v našem primeru enak. Samo če stroj deluje 100-odstotno za časovni interval ene minute, lahko rečemo, da deluje. Z vektorizirano učno množico in binariziranim klasifikacijskim razredom se lahko lotimo procesa učenja modelov.

Poglavje 4

Eksperimenti in rezultati

V eksperimentalnem delu bomo poskušali zgraditi model za napovedovanje delovanja stroja z uporabo prej opisanih algoritmov, metrik ter pristopov za filtriranje atributov.

4.1 Potek eksperimentalnega dela

Izvajanje eksperimentalnega dela smo razdelili na 3 razdelke, kjer smo uporabili različne pristope za reševanje problema napovedi okvar strojev.

V prvem razdelku se bomo lotili najbolj preprostega reševanja problema. Napovedne probleme bomo poskušali rešiti brez dodajanja zgodovinskih podatkov v učne vektorje in brez uporabe modelov za delo s časovnimi vrstami oziroma časovno odvisnimi podatki (logistična regresija, naključni gozdovi, metoda podpornih vektorjev). Cilj razdelka je ugotoviti, ali lahko s preprostejšimi pristopi strojnega učenja še vedno dobimo zadovoljive rezultate.

V drugem razdelku bomo iste modele trenirali z "obogateno" množico podatkov. Podatke bomo obogatili s povečanjem števila atributov. Vsak atribut v prvotni množici podatkov bomo razširili z n dodatnimi atributi, kjer n predstavlja število časovnih zamikov atributa v preteklost. Da zmanjšamo problemski prostor, smo za n določili konstanto 10, kar pomeni, da bomo iz vsakega atributa dobili 10 novih atributov. Celoten postopek bogatenja

podatkov bo podrobneje opisan v razdelku 4.4.

V zadnjem razdelku bomo uporabili gradnike LSTM in GRU, ki so primerni za delo s časovno odvisnimi podatki. Modele bomo trenirali s prvotnimi podatki, na katerih ne bomo izvajali procesa "bogatenja" podatkov, saj imajo modeli sposobnost pomnjenja in dodatno bogatanje podatkov nima smisla.

V vsakem razdelku bomo na izhodiščnih podatkih pognali algoritem Relief, s katerim bomo izbrali podmnožico atributov, ki bodo uporabljeni kot vhodni podatki v model. Filtrirano množico podatkov bomo nato razdelili na učno, testno in validacijsko množico. Na učni množici se bo izvajal proces učenja modela, na validacijski se bo izvajalo vrednotenje modela za potrebe izbire konfiguracijskih parametrov, testno množico pa bomo uporabili za testiranje uspešnosti naučenega modela.

Pri deljenju podatkov na učno, validacijsko in testno množico imamo več pristopov. Za namene diplomske naloge si bomo izbrali **stratificirano prečno preverjanje** (angl. stratified cross validation). 90 % vseh podatkov bomo vzeli kot učno-validacijsko množico, preostalih 10 % pa bomo uporabili kot testno množico. Podatke v učno-validacijski množici bomo nato razdelili na 10 enakomernih delov, kjer bomo v vsaki iteraciji prečnega preverjanja en del uporabili kot validacijsko množico, preostalih 9 pa kot učno množico. Pri deljenju začetnih podatkov na podmnožice uporabljamo postopek **stratificiranja**, kjer pazimo, da je razmerje med številom primerov s pozitivnim (ne_deluje) in negativnim (deluje) klasifikacijskim razredom v posamezni množici enako.

Pri treniranju modelov bomo uporabili pristop **mrežno iskanje** (angl. grid search). Namen pristopa je poskusiti različne kombinacije konfiguracijskih parametrov modelov in primerjati različne konfiguracije med seboj. Modele bomo v postopku učenja vrednotili na podlagi vnaprej določene metrike, v našem primeru je to **občutljivost**. Izhodni rezultat mrežnega iskanja je model z najvišjo vrednostjo izbrane metrike in najboljša kombinacija konfiguracijskih parametrov.

Vsak model ima lasten nabor parametrov, s katerimi lahko vplivamo na njegovo uspešnost. V spodnjem seznamu so opisani konfiguracijski parametri za vsak uporabljen model.

- Logistična regresija: Ne vsebuje nobenih konfiguracijskih parametrov, zato tukaj ne bomo izvajali mrežnega iskanja.

- Naključni gozdovi: Na voljo imamo dva konfiguracijska parametra *mtry* in *ntree*. Parameter *mtry* predstavlja velikost množice naključno izbranih atributov, ki jo uporabljamo pri grajenju vozlišč odločitvenih dreves. Atribut *ntree* pa predstavlja število odločitvenih dreves v naključnem gozdu.

- SVM: Pri algoritmu SVM imamo prav tako na voljo dva atributa, in sicer $C(cost)$ in σ. C predstavlja ceno napačne klasifikacije primerov. Večja kot je vrednost parametra C, strožje so meje, ki ločujejo oba razreda, oziroma bolj se model prilagaja učnim podatkom in slabše posplošuje. Parameter σ se uporablja pri konfiguraciji **radialne funkcije**, ki se uporablja za preslikavo problemskega prostora (radialna funkcija je specifičen primer jedrnih funkcij). Natančneje povedano, *sigma* je enaka standardnemu odklonu Gaussove krivulje, ki se uporablja pri radialni funkciji. V spodnji enačbi imamo zapisano formulo radialne funkcije:

$$k(x, x') = exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \tag{4.1}$$

- Pri LSTM in GRU ter nevronskih mrežah imamo na voljo ogromno konfiguracijskih parametrov. Da zmanjšamo kompleksnost preiskovanja parametrov, smo kot konfiguracijski parameter izbrali **število epoh**. Na kratko povedano, epoha je enaka trenutku v procesu učenja nevronske mreže, ko smo nevronski mreži izpostavili vse učne podatke. Večje kot je število epoh, daljši je čas treniranja in boljše se napovedna funkcija prilega učnim podatkom.

Uspešnost modela bomo nato ovrednotili in določili, ali je model uporaben za izdelavo napovednega sistema. Pri pogajanju določitve praga uporabnosti (minimalne mere uspešnosti modela) s sodelujočim podjetjem smo se odločili naslednje: če je mera občutljivosti modela višja od 70 %, bomo model upoštevali/obravnavali kot **uporaben**, v nasprotnem primeru pa kot **neuporaben**.

4.2 Programska oprema in okolje

Za opravljanje eksperimentalnega dela smo si izbrali statistični paket **R** in **RStudio** - to je grafični vmesnik ter zbirka programskih orodij za široko uporabljen jezik R. Ker je jezik zelo široko uporabljen na področju statične analize, strojnega učenja in umetne inteligence, je idealen za reševanje problema napovedi okvar strojev. **R** nam ponuja široko izbiro knjižnic, ki implementirajo razne algoritme in pristope strojnega učenja. Za potrebe diplomske naloge smo si izbrali naslednje knjižnice:

- **glm**: implementacija logistične regresije ter posplošenih linearnih modelov,
- **randomForest**: implementacija modela naključnega gozda,
- **kernlab**: implementacija modela metode podpornih vektorjev,
- **keras**: implementacija gradnikov LSTM in GRU oziroma nevronskih mrež,
- **caret**: zbirka orodij za manipulacijo podatkov ter delo z modeli za strojno učenje.

4.3 Napovedovanje delovanja brez uporabe zgodovinskih podatkov

4.3.1 Izbira parametrov

Ker uporabljamo postopek mrežnega iskanja, moramo pri vsakem modelu določiti zalogo vrednosti konfiguracijskih parametrov. Pri naključnih drevesih smo parameter *ntree* nastavili na konstantno vrednost $ntree = 200$, parametru *mtry* pa smo določili zalogo vrednosti $[3, 6, 9]$. Pri metodi pomožnih vektorjev smo parametru σ določili zalogo vrednosti $[0.1, 1.0, 3.0]$, parametru $C(cost)$ pa zalogo vrednosti $[0.75, 5.00, 10.00]$.

4.3.2 Izbira atributov

V tabeli 4.1 so prikazani rezultati Reliefa za vse tri obravnavane učne probleme.

Napovedni interval \Atribut	moxa_1	moxa_2	moxa_3	moxa_4	moxa_5	moxa_9	moxa_10	moxa_11	moxa_12
30 minut	0.021	0.007	0.002	0.015	0.111	0.010	0.002	0.018	0.008
1 ura	0.018	0.009	0.002	0.019	0.070	0.12	0.001	0.019	0.010
8 ur	0.011	0.009	0.002	0.005	0.042	0.008	0.001	0.006	0.004
Napovedni interval \Atribut	moxa_14	moxa_15	moxa_21	moxa_22	moxa_23	moxa_24	moxa_25	moxa_26	moxa_27
30 minut	0.009	0.019	0.007	0.009	0.043	0.005	0.003	0.017	0.003
1 ura	0.015	0.021	0.004	0.005	0.049	0.002	0.002	0.016	0.003
8 ur	0.004	0.013	0.003	0.003	0.005	-0.001	-0.001	0.006	0.002

Tabela 4.1: Ocene atributov z algoritmom Relief

Začetna množica podatkov vsebuje 18 atributov. Pri izbiri podmnožice atributov lahko zagotovo odstranimo tiste, ki imajo negativno oceno. Ker nam to pravilo problemskega prostora ne poenostavi dovolj, smo množico atributov dodatno zmanjšali tako, da smo izbrali le 10 najboljših atributov. V spodnjem seznamu so zapisane izbrane množice atributov za vsak napovedni problem:

- 30 minut: moxa_1, moxa_4, moxa_5, moxa_9, moxa_11, moxa_14, moxa_15, moxa_22, moxa_23, moxa_26,

- 1 ura: moxa_1, moxa_4, moxa_5, moxa_9, moxa_11, moxa_12, moxa_14, moxa_15, moxa_23, moxa_26,
- 8 ur: moxa_1, moxa_2, moxa_4, moxa_5, moxa_9, moxa_11, moxa_12, moxa_15, moxa_23, moxa_26,

Poleg ločenih ocen za vsak napovedni interval smo v tabeli 4.2 zgradili lestvico najboljše ocenjenih atributov za vse tri napovedne probleme v povprečju. Iz tabele 4.2 lahko razberemo, kateri fizikalni dejavniki (temperatura,

Atribut	povprečna ocena
moxa_5	$13,80 * 10^-2$
moxa_23	$3,23 * 10^-2$
moxa_15	$1,76 * 10^-2$
moxa_11	$1,43 * 10^-2$
moxa_1	$1,66 * 10^-2$
moxa_4	$1,30 * 10^-2$
moxa_26	$1,30 * 10^-2$
moxa_9	$1,00 * 10^-2$
moxa_14	$0,93 * 10^-2$
moxa_2	$0,83 * 10^-2$
moxa_12	$0,73 * 10^-2$
moxa_22	$0,56 * 10^-2$
moxa_21	$0,46 * 10^-2$
moxa_27	$0,26 * 10^-2$
moxa_24	$0,20 * 10^-2$
moxa_3	$0,20 * 10^-2$
moxa_10	$0,13 * 10^-2$
moxa_25	$0,13 * 10^-2$

Tabela 4.2: Povprečne vrednosti ocene algoritma Relief za vse tri napovedne probleme, urejene od najboljše do najslabše ocenjenega atributa

tlak) in katere komponente (grelne plošče, ventili) najbolj vplivajo na delovanje stroja. Če si ogledamo najboljše ocenjene atribute, lahko opazimo, da brizganje tekočine v kalup (moxa_23) bolj vpliva na delovanje stroja kot stanje ventila (moxa_10). Z dodatno analizo zgornjih podatkov bi lahko ugotovili tudi dodatne odvisnosti med delovanjem stroja in preostalih signalov.

V naslednjem razdelku bomo pognali proces učenja z izbranimi atributi in primerjali rezultate modelov med seboj.

4.3.3 Evalvacija modelov

Po izbiri atributov s pomočjo algoritma Relief smo se lotili učenja modelov. Tabele 4.3, 4.4 in 4.5 podajajo ocene uspešnosti naučenih modelov.

Napovedni interval: 30 minut

Model \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
LR	81,47 %	80,54 %	82.00 %	80,33 %	0,814
RF, $mtry = 3$	89,10 %	85,70 %	92,11 %	88,08 %	0,892
SVM, $\sigma = 3, C = 0.75$	86,31 %	85,18 %	87,31 %	85,40 %	0,863

Tabela 4.3: Napovedovanje delovanja stroja za 30 minut vnaprej

Tabela 4.3 prikazuje, da klasifikacijski model RF dosega najboljšo uspešnost. Ne samo, da prekaša preostale modele na področju **občutljivosti**, ampak jih prekaša tudi v preostalih metrikah.

Na drugem mestu imamo model SVM, ki se pri občutljivosti zelo približuje RF, na področju specifičnosti in F-ocene pa lahko opazimo večje razlike. To pomeni, da RF boljše zaznava negativne klasifikacijske razrede in da je napoved negativnega razreda točnejša.

Na zadnjem mestu imamo LR, ki kot pričakovano dosega najslabšo uspešnost, saj rešujemo nelinearen problem. Pri izbiri modela napovednega sistema bi se najbolj nagibali k RF, saj je njegova uspešnost v splošnem najboljša. Poleg tega je čas učenja pri RF občutno krajši kot pri SVM, kar je še dodaten razlog za izbiro tega modela. Če nam je hitrost največjega pomena, bi lahko izbrali LR. V splošnem bi za uporabo v praksi izbrali katerikoli model, saj vsi presegajo prej določen prag **uporabnosti**.

Model \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
Logistična regresija	70,69 %	66,75 %	74,17 %	68,12 %	0,706
Naključni gozd, $mtry = 6$	77,00 %	73,74 %	79,89 %	75,0 5%	0,769
Metoda podpornih vektorjev, $\sigma = 3, C = 0.75$	77,18 %	75,29 %	78,86 %	75,58 %	0,771

Tabela 4.4: Napovedovanje delovanja stroja za eno uro vnaprej

Napovedni interval: 1 ura

Če primerjamo tabeli 4.4 in 4.3, lahko opazimo, da v splošnem vsi modeli kažejo nižjo uspešnost. Za razliko od prejšnjega modela, kjer smo imeli RF na prvem mestu, je zdaj SVM zavzel mesto najuspešnejšega modela. Prav tako kot RF v prejšnjem razdelku je SVM dosegel boljšo uspešnost, s to razliko, da so razlike v uspešnosti modelov manjše.

Drugo mesto zavzema model RF. V primerjavi s prejšnjim modelom nam je tukaj postopek mrežnega iskanja izbral višjo vrednost konfiguracijskega parametra *mtry*, kar kaže na višanje kompleksnosti problema.

Nazadnje imamo še model LR, ki ni dosegel praga **uporabnosti** in ga zato ne moremo uporabiti v napovednem sistemu. Pri izbiri klasifikatorja za napovedni model bi se odločali med modeloma SVM in RF. Razlika v uspešnosti modela je v tem primeru občutno manjša, kar odraža razlika vrednosti AUC. Višja mera občutljivosti in F-ocene pa kažejo na boljšo sposobnost klasifikacije pozitivnih razredov in višjo stopnjo verodostojnosti napovedi pri modelu SVM. Boljša uspešnost modela SVM nam pri tako nizki občutljivosti pomeni več kot hitrost RF, zato si bomo kot napovedni model izbrali SVM.

Napovedni interval: 8 ur

Uspešnost klasifikacijskih modelov je drastično upadla v primerjavi s prejšnjimi napovednimi problemi. V tabeli 4.5 lahko opazimo, da noben klasifikacijski model ne dosega praga **uporabnosti**. Na podlagi mere AUC in občutljivosti lahko opazimo, da nam najboljšo uspešnost ponuja RF, vendar ne dosega praga uporabnosti in ga zato ne moremo uporabiti kot napovedni

Model \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
Logistična regresija	59,28 %	42,65 %	74,55 %	50,08 %	0,596
Naključni gozd, $mtry = 6$	69,12 %	64,89 %	73,00 %	66,80 %	0,691
Metoda podpornih vektorjev, $\sigma = 5, C = 1$	65,19 %	54,23 %	75,27 %	59,87 %	0,655

Tabela 4.5: Napovedovanje delovanja stroja za 8 ur vnaprej

model.

Na podlagi zgornjih ugotovitev lahko trdimo, da problem z napovednim intervalom 8 ur ni zadovoljivo rešljiv, ker noben od izbranih modelov ne kaže zadovoljive uspešnosti. Eden izmed potencialnih razlogov za slabo uspešnost modelov je kratek interval vzorčenja (trenutno je ena minuta) ali pomanjkanje sposobnosti pomnjenja pri uporabljenih modelih. V naslednjem razdelku bomo skušali izboljšati uspešnost modela z uporabo širšega vzorčnega intervala 10 minut, ki ga bomo zgradili z uporabo enominutnih intervalov.

4.4 Napovedovanje delovanja z uporabo zgodovinskih podatkov

4.4.1 Priprava podatkov

V prejšnjem razdelku smo se reševanja problema lotili tako, da smo na osnovi enominutnih intervalov napovedovali delovanje stroja za n časovnih intervalov vnaprej. Rezultati so bili zadovoljivi, vendar je kakovost modela hitro upadala z večanjem napovednega intervala.

V tem razdelku smo se reševanja problema lotili tako, da smo prvotne vhodne podatke obogatili z zgodovinskimi vrednostmi posameznih atributov. Recimo, da si kot primer vzamemo atribut moxa_1. Vhodni vektor bi razširili tako, da bi ustvarili nove atribute moxa_1_t, kjer t predstavlja indeks zamaknjene vrednosti atributa. Če bi želeli vektor razširiti za 10 minut, bi bila zaloga vrednosti t enaka [1,...,10], kjer je $t = 10$ najnovejša vrednost

vektorja, $t = 1$ pa najstarejša vrednost vektorja. Razširjanje atributa smo implementirali v funkciji, ki je prikazana v algoritmu 2.

Algoritem 2: Psevdokoda funkcije agregate_samples

```
Generiranje enakomernih časovnih intervalov;
Inicializacija izhodne matrike;
Inicializacija izhodnega vektorja;
foreach časovni interval do
    V izvorni matriki poišči vrednosti, ki spadajo v trenutni časovni
     interval;
    Najdene vrednosti vstavi na ustrezno mesto v izhodnem vektorju;
    if Manjkajoče vrednosti v vektorju then
        continue
    else
        Izhodni vektor vstavi v izhodno matriko;
```

V našem primeru smo izhodiščno matriko obogatili s podatki za **10 minut** nazaj. Nova matrika tako vsebuje 180 atributov (izhodiščna matrika ima 18 atributov, za vsak atribut 10 zgodovinskih vrednosti, kar nas privede do 180 atributov).

4.4.2 Izbira atributov

Novo ustvarjene podatke smo nato pognali skozi algoritem Relief. Ker imamo pri trenutnem pristopu 10-krat več podatkov, bi bilo prikazovanje ocen vseh atributov zelo nepregledno. Zato v tabelah 4.6 - 4.8 prikazujemo **20** najboljše ocenjenih atributov.

Čas \Atribut	moxa_5_1	moxa_5_2	moxa_5_3	moxa_5_4	moxa_5_5	moxa_5_6	moxa_5_7	moxa_5_8	moxa_5_9	moxa_5_10
30 minut	$25,4 * 10^{-2}$	$25,2 * 10^{-2}$	$26,2 * 10^{-2}$	$27,0 * 10^{-2}$	$29,2 * 10^{-2}$	$29,4 * 10^{-2}$	$29,3 * 10^{-2}$	$29,0 * 10^{-2}$	$25,7 * 10^{-2}$	$30,0 * 10^{-2}$
Čas \Atribut	moxa_4_3	moxa_4_4	moxa_4_5	moxa_4_6	moxa_4_7	moxa_4_9	moxa_23_1	moxa_23_4	moxa_23_7	moxa_23_9
30 minut	$6,00 * 10^{-3}$	$5,85 * 10^{-3}$	$5,28 * 10^{-3}$	$5,83 * 10^{-3}$	$6,13 * 10^{-3}$	$5,69 * 10^{-3}$	$5,34 * 10^{-3}$	$5,79 * 10^{-3}$	$5,96 * 10^{-3}$	$5,89 * 10^{-3}$

Tabela 4.6: Rezultati algoritma Relief za 30 minut vnaprej

Čas \Atribut	moxa_5_1	moxa_5_2	moxa_5_3	moxa_5_4	moxa_5_5	moxa_5_6	moxa_5_7	moxa_5_8	moxa_5_9	moxa_5_10
30 minut	$14,3 * 10^{-2}$	$13,0 * 10^{-2}$	$13,3 * 10^{-2}$	$14,2 * 10^{-2}$	$14,4 * 10^{-2}$	$13,9 * 10^{-2}$	$14,0 * 10^{-2}$	$14,9 * 10^{-2}$	$15,8 * 10^{-2}$	$14,6 * 10^{-2}$
Čas \Atribut	moxa_4_3	moxa_4_4	moxa_4_8	moxa_23_1	moxa_23_2	moxa_23_3	moxa_23_4	moxa_23_6	moxa_23_7	moxa_23_10
30 minut	$5,47 * 10^{-2}$	$6,4 * 10^{-2}$	$5,87 * 10^{-2}$	$5,67 * 10^{-2}$	$5,43 * 10^{-2}$	$5,33 * 10^{-2}$	$5,71 * 10^{-2}$	$5,53 * 10^{-2}$	$6,08 * 10^{-2}$	$5,69 * 10^{-2}$

Tabela 4.7: Rezultati algoritma Relief za 1 uro vnaprej

Čas \Atribut	moxa_5_1	moxa_5_2	moxa_5_3	moxa_5_4	moxa_5_5	moxa_5_6	moxa_5_7	moxa_5_8	moxa_5_9	moxa_5_10
30 minut	$5,11 * 10^{-2}$	$5,04 * 10^{-2}$	$5,52 * 10^{-2}$	$5,33 * 10^{-2}$	$5,52 * 10^{-2}$	$4,58 * 10^{-2}$	$4,80 * 10^{-2}$	$5,17 * 10^{-2}$	$4,92 * 10^{-2}$	$4,36 * 10^{-2}$
Čas \Atribut	moxa_4_1	moxa_4_9	moxa_4_10	moxa_11_6	moxa_11_7	moxa_11_10	moxa_23_10	moxa_24_2	moxa_24_10	moxa_25_10
30 minut	$3,21 * 10^{-2}$	$2,40 * 10^{-2}$	$2,54 * 10^{-2}$	$3,61 * 10^{-2}$	$2,43 * 10^{-2}$	$4,42 * 10^{-2}$	$2,37 * 10^{-2}$	$3,74 * 10^{-2}$	$2,61 * 10^{-2}$	$2,82 * 10^{-2}$

Tabela 4.8: Rezultati algoritma Relief za 8 ur vnaprej

Čeprav imamo pri trenutnem pristopu veliko več atributov, se v rezultatih ponavljajo isti izstopajoči atributi kot pri prejšnjem pristopu. Prvih 10 najboljših mest zavzemajo zgodovinske vrednosti atributa moxa_5, preostalih 10 mest pa si po večini delita atribut moxa_23, ki je bil v prejšnjem razdelku drugi najboljše ocenjen atirbut, in moxa_4, ki spada pod 10 najboljših atributov iz prejšnjega razdelka.

Pri trenutnem pristopu smo poleg uporabe drugačne učne množice povečali število izbranih atributov z 10 na 20. Glavni razlog za povečanje števila atributov je raznolikost množice atribtov. Če si ogledamo zgornje tabele, lahko opazimo, da prvih 10 mest vedno zasedajo zgodovinske vrednosti atributa moxa_5. Če bi vzeli samo 10 najboljše ocenjenih atributov, bi pomenilo, da skušamo delovanje stroja napovedati samo z uporabo atributa moxa_5. Z razširitvijo končne množice atributov smo v napovedovanje okvar stroja vključili večje število različnih atributov.

Za lažje razumevanje zgornje razlage smo v spodnjem seznamu prikazali imena 20 najboljših atributov za posamezen napovedni problem:

- 30 minut: moxa_5_1, moxa_5_2, moxa_5_3, moxa_5_4, moxa_5_5, moxa_5_6, moxa_5_7, moxa_5_8, moxa_5_9, moxa_5_10, moxa_4_3, moxa_4_4, moxa_4_5, moxa_4_7, moxa_4_9, moxa_23_1, moxa_23_4, moxa_23_7, moxa_23_9 ,

- 1 ura: moxa_5_1, moxa_5_2, moxa_5_3, moxa_5_4, moxa_5_5, moxa_5_6,

moxa_5_7, moxa_5_8, moxa_5_9, moxa_5_10, moxa_4_3, moxa_4_4, moxa_4_8, moxa_23_1, moxa_23_2, moxa_23_3, moxa_23_4, moxa_23_6, moxa_23_7, moxa_23_10,

- 8 ur: moxa_5_1, moxa_5_2, moxa_5_3, moxa_5_4, moxa_5_5, moxa_5_6, moxa_5_7, moxa_5_8, moxa_5_9, moxa_5_10, moxa_4_1, moxa_4_9, moxa_4_10, moxa_11_6, moxa_11_7, moxa_11_10, moxa_23_10, moxa_24_2, moxa_24_10, moxa_25_10.

Z uporabo zgornjega seznama in tabel smo zgradili tabelo 20 v povprečju najboljše ocenjenih atributov za vse tri napovedne probleme. Tabela 4.9 vsebuje povprečne ocene posameznih atributov, urejene od najboljše do najslabše ocenjenega atributa.

Atribut	povprečna ocena
moxa_5_9	$16,83 * 10^-2$
moxa_5_8	$16,40 * 10^-2$
moxa_5_5	$16,39 * 10^-2$
moxa_5_10	$16,35 * 10^-2$
moxa_5_7	$16,06 * 10^-2$
moxa_5_6	$15,98 * 10^-2$
moxa_5_4	$15,54 * 10^-2$
moxa_5_3	$14,99 * 10^-2$
moxa_5_1	$14,93 * 10^-2$
moxa_5_2	$14,44 * 10^-2$
moxa_4_8	$5,56 * 10^-2$
moxa_4_3	$4,37 * 10^-2$
moxa_11_10	$4,36 * 10^-2$
moxa_4_4	$4,23 * 10^-2$
moxa_23_10	$4,22 * 10^-2$
moxa_4_7	$4,02 * 10^-2$
moxa_23_7	$3,95 * 10^-2$
moxa_4_9	$3,88 * 10^-2$
moxa_23_1	$3,84 * 10^-2$
moxa_4_1	$3,80 * 10^-2$

Tabela 4.9: Povprečne vrednosti ocene algoritma Relief za vse tri napovedne probleme, urejene od najboljše do najslabše ocenjenega atributa

Najboljše ocenjen atribut prejšnjega pristopa ima vrednost blizu $14,00 *$

10^-2, tukaj pa ima vrednost $16,00 * 10^-2$. Tudi na splošno so v trenutnem pristopu atributi boljše ocenjeni kot v prejšnjem pristopu. To lahko nakazuje na boljšo uspešnost klasifikacijskih modelov, saj trenutni vhodni podatki vsebujejo več informacij kot pri prejšnjem pristopu. V naslednjih razdelkih se bomo lotili treniranja modelov, kar bo odgovorilo na zgoraj zastavljeno vprašanje.

4.4.3 Izbira parametrov

Ko smo izbrali končne atribute za učenje modela, moramo izbrati množico parametrov, nad katerimi bomo izvajali mrežno iskanje. Ker učna množica vsebuje večje število atributov, moramo prilagoditi zalogo vrednosti parametrov, ki se uporabljajo pri mrežnem iskanju.

Pri RF smo zalogo vrednosti parametra *mtry* povečali s [3,6,9] na [3,6,9, 13,16,20], vrednost atributa *ntree* pa pustili na konstanti vrednosti 200. Zalogo vrednosti za SVM smo pustili enako, saj za razliko od RF parametri niso toliko odvisni od števila atribtov. Če bi pri naključnih gozdovih pustili enako zalogo vrednosti, bi klasifikacijski model omejili na izbiro 50 % vseh atributov, ki so na voljo.

4.4.4 Evalvacija modelov

Napovedni interval: 30 minut

Zamik \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
Logistična regresija	78,66 %	72,37 %	84,09 %	75,86 %	0,788
Naključni gozd, $mtry = 6$	84,15 %	77,63 %	89,77 %	81,94 %	0,845
Metoda podpornih vektorjev, $\sigma = 0.1, C = 0.75$	83,54 %	80,26 %	86,36 %	81,88 %	0,835

Tabela 4.10: Napovedovanje delovanja stroja za 30 minut vnaprej

Če si ogledamo rezultate v tabelah 4.3 in 4.10, lahko opazimo, da trenutni modeli v splošnem kažejo slabšo uspešnost. Eden od razlogov je lahko

povečanje dimenzionalnosti vhodnih podatkov. Ko smo povečali število atributov, nismo samo obogatili vhodnih podatkov, temveč smo tudi povečali kompleksnost napovedovanja. V postopek minimizacije funkcije napake smo dodali enkrat večje število odvisnih spremenljivk. Drugi razlog je lahko manjša variabilnost atributov v podatkih. Čeprav imajo uporabljeni atributi višjo povprečno oceno, to ni neposredni kazalnik za višjo uspešnost modelov. V primerjavi s prejšnjim pristopom tukaj po večini uporabljamo zgodovinske vrednosti največ 4 atributov, kar je manjše od prejšnjih 18. Na podlagi trenutnih rezultatov se ne moremo odločiti, ali obogatenje učne množice podatkov pomaga pri izboljšavi uspešnosti klasifikacijskih modelov.

Pri izbiri napovednega modela bi si v tem primeru izbrali SVM, saj dosega najvišjo mero občutljivosti. Če sta nam hitrost učenja ali razumljivost modela večjega pomena, ustrezata tudi druga dva modela, saj oba presegata prag **uporabnosti**.

Napovedni interval: 1 ura

Zamik \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
Logistična regresija	74,39 %	71,05 %	77,27 %	72,00 %	0,743
Naključni gozd, $mtry = 9$	79,27 %	72,37%	85,23 %	76,39 %	0,795
Metoda podpornih vektorjev, $\sigma = 3, C = 0.75$	79,27 %	76,32 %	81,82 %	77,33 %	0,792

Tabela 4.11: Napovedovanje delovanja stroja za eno uro vnaprej.

Če primerjamo uspešnost LR v tabelah 4.4 in 4.11, lahko opazimo, da pristop brez uporabe zgodovinskih podatkov izkazuje nižjo uspešnost od trenutnega pristopa. To je nepričakovano, saj bi po rezultatih prejšnjega napovednega problema sklepali, da bo trenuten pristop kazal nižjo uspešnost. Prav tako kot LR sta tudi preostala dva modela kazala višjo uspešnost, čeprav je v primeru RF vrednost občutljivosti slabša. Boljši rezultati pri trenutnem napovednem problemu še ne pomenijo, da je ta pristop uporabe zgodovinskih podatkov boljši. Pri naslednjem napovednem problemu bomo videli, ali je

pristop uporabe zgodovinskih podatkov dejansko boljši ali je zgoraj opisana uspešnost posledica naključja, ki ga povzročijo specifike uporabljene metode in napovednega problema. Pri trenutnem pristopu in napovednem problemu vsi modeli zadostujejo pragu **uporabnosti**. Pri izbiri napovednega modela si bomo izbrali SVM, saj ima najvišjo uspešnost vseh treh modelov.

Napovedni interval: 8 ur

Zamik \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
Logistična regresija	53,12 %	34,21 %	70,24 %	40,94 %	0,525
Naključni gozd, $mtry = 13$	70,00 %	64,47 %	75,00 %	67,12 %	0,700
Metoda podpornih vektorjev, $\sigma = 1, C = 10$	60,00 %	47,37 %	71,43 %	52,94 %	0,600

Tabela 4.12: Napovedovanje delovanja stroja za 8 ur vnaprej.

Če primerjamo tabeli 4.5 in 4.12, lahko opazimo, da imamo pri napovednem intervalu 30 minut imamo z uporabo zgodovinskih podatkov slabšo uspešnost kot pri pristopu brez njihove uporabe. V primeru LR in SVM bi dosegli boljše rezultate, če bi imeli klasifikator, ki napoveduje samo večinski razred. V primeru RF je uspešnost modela nižja od pristopa brez uporabe zgodovinskih podatkov, vendar še vedno nezadovoljiva. Noben od zgoraj uporabljenih modelov ne dosega **praga uporabnosti**, kar pomeni, da napovedni problem ni zadovoljivo rešljiv.

4.5 Napovedovanje delovanja z nevronskimi mrežami

V prejšnjem razdelku smo modelom brez sposobnosti pomnjenja odstranili to pomanjkljivost tako, da smo učno množico obogatili z zgodovinskimi podatki. V trenutnem razdelku smo se reševanja problema lotili z uporabo **rekurenčnih nevronskih mrež** oziroma gradnikov LSTM in GRU, ki imata

sposobnost pomnjenja, kar pomeni, da pri trenutnem pristopu ne bomo uporabljali metode bogatenja vhodnih podatkov. Pri filtriranju atributov bomo uporabili tabelo 4.1, ki smo jo uporabili v razdelku 4.3.

4.5.1 Izbira parametrov

V primerjavi s prej uporabljenimi modeli (LR, RF, SVM) pri učenju nevronskih mrež uporabljamo postopek **vzvratnega razširjanja napake** (angl. back-propagation). Postopek je kombinacija uporabe **verižnega pravila** (angl. chain rule) in **gradientnega spusta** (angl. gradient descent), ki se uporablja za izračun uteži v nevronskih mrežah [14]. Učenje poteka v iteracijah, kjer ob vsaki iteraciji uporabimo podmnožico učne množice za izračun uteži. Velikost podmnožice v posamezni iteraciji je podana s parametrom **velikost paketa** (angl. batch size). Ob vsaki novi iteraciji uporabimo podatke, ki niso bili uporabljeni v prejšnjih iteracijah. Trenutku, ko smo uporabili vse podatke učne množice, pravimo **epoha** (angl. epoch). Ta predstavlja iteracijo, kjer smo nevronski mreži izpostavili vse učne primere. Večje kot je število epoh, daljši je čas učenja in boljše je prilagajanje (angl. fitting) nevronske mreže na učne podatke.

Za izvajanje eksperimenta smo si izbrali naslednje vrednosti parametrov:

- velikost paketa = 10,
- število nevronov v posamezni plasti nevronskih mrež = 50,
- število epoh = 10, če model v tem času ne uspe najti minimalne napake, bomo število epoh povečali.

4.5.2 Vrste gradnikov

Gradniki LSTM in GRU obstajajo v dveh oblikah, in sicer:

- z ohranjanjem notranjega stanja (angl. stateful) in
- brez ohranjanja notranjega stanja (angl. stateless).

Gradniki LSTM/GRU imajo spominsko celico, ki je zadolžena za hranjenje pomembnih podatkov. Pri gradnikih brez ohranjanja notranjega stanja se notranje stanje gradnika ponastavi ob vsakem novem paketu podatkov, medtem ko se stanje v celicah z ohranjanjem notranjega stanja ohranja skozi celotno iteracijo epohe. Glavna razlika med vrstami gradnikov je v dolžini spomina. Gradniki z ohranjanjem notranjega stanja si lahko zapomnijo daljša zaporedja in s tem boljše modelirajo podatke, ki so časovno odvisni. Gradniki brez ohranjanja notranjega stanja pa imajo krajši spomin in si tako lažje zapomnijo krajša zaporedja, iz česar sklepamo, da bodo na podlagi rezultatov iz razdelka 4.4 delovali boljše kot gradniki z ohranjanjem notranjega stanja. V razdelku 4.5.3 bomo primerjali uspešnost posameznih vrst gradnikov.

4.5.3 Evalvacija modelov

Napovedni problem: 30 minut

Model \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
LSTM brez ohranjanja stanja *epohe* = 10	95,75 %	88,15 %	98,83 %	92,28 %	0,961
LSTM z ohranjanjem stanja *epohe* = 50	93,92 %	87,86 %	96,37 %	89,28 %	0,929
GRU brez ohranjanja stanja *epohe* = 10	95,08 %	88,15 %	97,89 %	91,18 %	0,948
GRU z ohranjanjem stanja *epohe* = 10	96,08 %	88,15 %	99,30 %	92,85 %	0,967

Tabela 4.13: Napovedovanje delovanja stroja za 30 minut vnaprej z uporabo nevronskih mrež

Kot lahko opazimo iz tabele 4.13, so rezultati primerljivi. Če si ogledamo model LSTM, lahko opazimo, da model brez ohranjanja stanja dosega boljšo uspešnost v manjšem številu epoh. Vidimo, da moramo v tem primeru model LSTM z ohranjanjem stanja učiti dlje časa, da bi dosegli primerljive rezultate kot pri modelu LSTM brez ohranjanja stanja. Razlike pri gradnikih GRU so

manj očitne oziroma so v nasprotju z ugotovitvami kot pri gradnikih LSTM. V primerjavi s pristopoma iz razdelkov 4.3 in 4.4 imamo tu veliko boljše rezultate. Pri občutljivosti smo povprečno pridobili približno 3 %, pri točnosti in specifičnosti pa približno 10 %. Vsi modeli presegajo prag **uporabnosti**. Pri izbiri napovednega modela bi bilo najboljše, da si izberemo model GRU z ohranjanjem stanja, saj dosega najvišjo uspešnost. V naslednjem razdelku nadaljujemo z obravnavo problema z večjim napovednim intervalom.

Napovedni problem: 1 ura

Zamik \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
LSTM brez ohranjanja stanja *epohe* = 20	92,42 %	81,12 %	97,57 %	87,02 %	0,928
LSTM z ohranjanjem stanja *epohe* = 50	89,08 %	65,96 %	99,64 %	79,11 %	0,926
GRU brez ohranjanja stanja *epohe* = 20	92,67 %	81,12 %	97,94 %	87,39 %	0,933
GRU z ohranjanjem stanja *epohe* = 100	86,92 %	82,18 %	89,08 %	79,74 %	0,845

Tabela 4.14: Napovedovanje delovanja stroja za 1 uro vnaprej z uporabo nevronskih mrež

V tabeli 4.14 so razlike med modeli z ohranjanjem stanja in brez tega že bolj razvidne. Pri LSTM je razlika v specifičnosti kar velika, pri modelu GRU pa so razlike v uspešnosti zanemarljive. Poleg razlik v uspešnosti je LSTM brez ohranjanja stanja dosegel boljše rezultate v manjšem številu epoh. Podoben argument bi lahko uporabili tudi pri gradnikih GRU, kjer smo za doseganje podobne uspešnosti potrebovali manjše število epoh. Če naša predpostavka drži, so rezultati smiselni, saj gradnik z daljšim spominom potrebuje več časa za prilagoditev na krajša zaporedja, medtem ko se gradniku s krajšim spominom ni treba dodatno prilagajati.

V primerjavi s prejšnjimi pristopi so rezultati znova boljši. Pri napovednem problemu 1 ure smo pri prejšnjih pristopih imeli povprečno občutljivost

okoli 72 %, pri trenutnem pristopu pa je približno 81 %, če izločimo LSTM z ohranjanjem stanja. Poleg skoraj 10 % izboljšave občutljivosti so višje tudi preostale mere uspešnosti. Vsi modeli razen LSTM brez ohranjanja stanja presegajo prag **uporabnosti**. Pri izbiri napovednega modela bi se odločali med LSTM in GRU brez ohranjanja stanja, saj sta z veliko manjšim številom epoh dosegla podobno uspešnost kot GRU z ohranjanjem stanja. Na podlagi dosedanjih rezultatov lahko pri napovednem problemu 8 ur pričakujemo boljše rezultate kot z že preizkušenimi pristopi.

Napovedni problem: 8 ur

Zamik \Mera	Točnost	Občutljivost	Specifičnost	F-ocena	AUC
LSTM brez ohranjanja stanja *epohe* = 50	37,00 %	1,05 %	99,09 %	2,07 %	0,517
LSTM z ohranjanjem stanja *epohe* = 200	37,08 %	1,71 %	98,18 %	3,32 %	0,492
GRU brez ohranjanja stanja *epohe* = 50	36,92 %	00,06 %	99,54 %	01,30 %	0,541
GRU z ohranjanjem stanja *epohe* = 100	49,17 %	20,39 %	98,8 6%	33,70 %	0,693

Tabela 4.15: Napovedovanje delovanja stroja za 8 ur vnaprej z uporabo nevronskih mrež

Iz tabele 4.15 lahko povzamemo, da so rezultati napovednega problema 8 ur nezadovoljivi. Vsi modeli dosegajo slabšo uspešnost kot katerikoli model iz prejšnjih pristopov. Poleg tega se noben model ni niti približal pragu **uporabnosti**.

Poglavje 5

Sklep

V diplomski nalogi smo uporabili različne metode strojnega učenja za reševanje problema napovedovanja delovanja stroja za določen časovni interval vnaprej. Učenja modelov smo se lotili z uporabo različnih pristopov. Na modelih brez sposobnosti pomnjenja smo dodatno obogatili učne podatke in uporabili modele, kot sta LSTM in GRU, ki sta bila zasnovana za delo s časovno odvisnimi podatki. Na podlagi rezultatov smo ugotovili, kateri modeli so primernejši za reševanje posameznih napovednih problemov, katere vrednosti parametrov dajejo najboljše rezultate za posamezen model, kateri napovedni problemi so sploh rešljivi in kakšna je kompleksnost posameznih napovednih problemov.

5.1 Zaključne ugotovitve

Pri implementaciji in vrednotenju sistema za napovedovanje odpovedi stroja smo ugotovili:

1. **Uporaba zgodovinskih podatkov ne izboljša uspešnosti modelov**. Kot smo videli v razdelku 4.4, nam dodajanje zgodovinskih vrednosti atributov ne izboljša uspešnosti modelov. Glavni razlog za to je, da določeni modeli niso namenjeni delu s časovno odvisnimi podatki.

2. **Dobra uspešnost modelov GRU in LSTM kaže na obstoj zaporedij in časovnih odvisnosti v izvornih podatkih**. To pomeni, da je delovanje stroja za določen časovni interval vnaprej odvisno tudi od njegovega minulega delovanja.

3. **Izvorni podatki vsebujejo krajša časovno zaporedja**. V razdelku 4.5 smo ugotovili, da imata modela GRU in LSTM s krajšim spominom boljšo uspešnost kot modeli z daljšim spominom.

4. **Napoved delovanja stroja za 8 ur vnaprej je pretežek problem**. Slaba uspešnost modelov pri napovednem problemu 8 ur v razdelkih 4.3, 4.4 in 4.5 kaže, da je problem z uporabljenimi pristopi težko obvladljiv. Vzroki so lahko preprosti modeli/konfiguracije ali preprosto prešibke povezave med atributi in klasifikacijskim razredom.

5.2 Nadaljnje delo

1. **Uporaba avtoregresijskih modelov za bogatenje učne množice**. V diplomski nalogi smo na kratko omenili pristop ARIMA, ki je ena izmed metod za napovedovanje časovnih vrst in je bil uporabljen v enem izmed člankov, omenjenih v sorodnih delih [1]. Tako kot so v sorodnem delu uporabili pristop ARIMA, bi lahko pri reševanju našega problema uporabili pristop **VAR** oziroma **VARIMA**, ki sta posebni različici avtoregresijskih modelov, namenjeni za delo z vektorji. Modela bi uporabili za generiranje dodatnih atributov (podobno kot v razdelku 4.4), s katerimi bi informacijsko obogatili učno množico podatkov.

2. **Napovedovanje z metodo učenja ansamblov**. Uporabili bi lahko isti pristop, kot ga uporablja RF, vendar bi namesto odločitvenih dreves uporabili različne kombinacije modelov. Za začetek bi lahko zgradili ansambel iz že uporabljenih modelov. V primeru izboljšav bi lahko določene modele zamenjali ali uporabili več istih. Pri uporabi metode

ansamblov se moramo zavedati, da z dodajanjem dodatnih modelov večamo kompleksnost modela, kar zahteva dodatno računsko moč.

3. **Uporaba drugačnega načina vzorčenja podatkov**. Trenutno smo v razdelku 3.2 uporabili povprečje za generiranje metrik. S tem izgubimo veliko informacij, še posebej z večanjem vzorčnega intervala. Namesto povprečja bi lahko uporabili druge metrike, kot so: standardna deviacija, minimum, maksimum, mediana ... Še boljše pa bi bilo, če bi uporabili vse, saj imamo tako za vzorčeno obdobje na voljo čim več informacij [1].

Literatura

[1] Marcia Baptista, Shankar Sankararaman, Ivo P de Medeiros, Cairo Nascimento Jr, Helmut Prendinger, and Elsa MP Henriques. Forecasting fault events for predictive maintenance using data-driven techniques and arma modeling. *Computers & Industrial Engineering*, 115:41–53, 2018.

[2] Giuseppe Bonaccorso. *Machine Learning Algorithms*. Packt Publishing, 2017.

[3] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555:1–9, 2014.

[5] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[6] Tin Kam Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE, 1995.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Grega Kešpret. Diplomska naloga: Estimation of prediction reliabilities in regression modelling of data streams, 5 2012.

[9] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier, 1992.

[10] Nikolaos Kolokas, Thanasis Vafeiadis, Dimosthenis Ioannidis, and Dimitrios Tzovaras. Forecasting faults of industrial equipment using machine learning classifiers. *2018 IEEE International Conference on INnovations in Intelligent SysTems and ApplicationsAt: Thessaloniki, Greece*, pages 1–7, 07 2018.

[11] Jaroslaw Kurek and Stanislaw Osowski. Support vector machine for fault diagnosis of the broken rotor bars of squirrel-cage induction motor. *Neural Computing and Applications*, 19(4):557–564, 2010.

[12] Brett Lantz. *Machine Learning with R.* Packt Publishing, 2015.

[13] Mostafa Larky, Mohammad Riahi, Erfan Ahadi, and Hamidreza Javidrad. Utilization of neural network in prognostics for industrial equipment. *mostafa*, pages 1–9, 01 2018.

[14] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[15] Teja Roštan. Magistrsko delo: Time series forecasting with long short-term memory neural networks. Master's thesis, University of Ljubljana, Faculty of Computer and Information Science, 6 2018.

[16] Weishan Zhang, Wuwu Guo, Xin Liu, Yan Liu, Jiehan Zhou, Bo Li, Qinghua Lu, and Su Yang. LSTM-based analysis of industrial iot equipment. *IEEE Access*, 6:23551–23560, 2018.